This Page is Inserted by IFW Indexing and Scanning Operations and is not part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☐ FADED TEXT OR DRAWING
- ☐ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☐ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: _____

IMAGES ARE BEST AVAILABLE COPY.
As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.

Google

Web Images Groups News Froogle more »

D Language JCore Search Advanced Search Preferences

Web

Results **11** - **20** of about **166** for **D Language JCore**. (**0.25** seconds)

[PDF] AspectJ™ Specification
File Format: PDF/Adobe Acrobat - View as HTML
... join points between **JCore** and the aspect **languages** are over ... 3.2) associate coordination schemes with **JCore** classes and ... in Figure 2. 3.1 I MPORT **D** ECLARATION A ...
gsd.web.elte.hu/Temak/Aop/specification.pdf - Similar pages

[PDF] Assessing Aspect Oriented Programming Preliminary Results
File Format: PDF/Adobe Acrobat - View as HTML
... r E li sa L A Ba niassa **d** a nd ... was to learn whether the com bination of **JCore** for component programming and COOL as a synchronization aspect **language** eased the ...
www.cs.ubc.ca/labs/se/papers/superceded/aop-assess.pdf - Similar pages

[PDF] An Initial Assessment of Aspect-oriented Programming
File Format: PDF/Adobe Acrobat - View as HTML
... Our hypothesis was that programmers working with the aspect-oriented programming **language**, AspectJ (**JCore** and Cool components), would be able to more quickly ...
www.cs.ubc.ca/labs/se/papers/1999/icse99-aop.pdf - Similar pages
[More results from www.cs.ubc.ca]

[PDF] Towards a pragmatic composition model of Corba services based on ...
File Format: PDF/Adobe Acrobat - View as HTML
... The aspect oriented programming **language D** [21] introduces three specific **languages**: **Jcore** expresses the basic functionality of the system, Ridl deals with ...
rainbow.essi.fr/publications/00-dea-bussard.pdf - Similar pages

[PDF] Aspect Oriented Logic Meta Programming Introduction TyRuBa
File Format: PDF/Adobe Acrobat - View as HTML
... public boolean empty return pos method **JCore** BoundedStack boolean ... reader is referred to our Ph **D** dissertation DV ... that part of the aspect **language** can be ...
trese.cs.utwente.nl/aop-ecoop98/papers/DeVolder.pdf - Similar pages

[PDF] JCORE (31-40)
File Format: PDF/Adobe Acrobat - View as HTML
... these issues has changed somewhat, the whole **language** of 'diversi ... A NTONY L ERMAN : I'**d** like to pick up ... I've been involved with **JCORE** since its early DEB ...
www.jcore.org.uk/downloads/RACE_RELATIONS.pdf - Similar pages

[PDF] ASPECT ORIENTEDPROGRAMMING
File Format: PDF/Adobe Acrobat - View as HTML
... The **language D** contained three components, a **language** for defining components (like Java or C) and ... The component definition **language** was **Jcore**, which was a ...
www.cs.virginia.edu/~techrep/CS-99-29.pdf - Similar pages

[PDF] Aspect Oriented Logic Meta Programming
File Format: PDF/Adobe Acrobat - View as HTML
... generation in our recent Ph **D** dissertation DVThestructureofthepaperisasfollows ... Java source code Named after **JCore** the simpli ed Java **language** which is ...
prog.vub.ac.be/Publications/1999/vub-prog-tr-99-05.pdf - Similar pages

A-Infos (en) Pull the Plug on the BNP - Postal workers in ...
... The last 100 posts, according to **language** Castellano_ Català_ ... Dr Richard Stone, chair of **JCORE** and member of the ... majordomo@tao.ca REPLIES: a-infos-**d**@tao.ca ...
www.ainfos.ca/99/may/ainfos00149.html - 10k - Cached - Similar pages

PORTAL
US Patent & Trademark Office

Subscribe (Full Service) Register (Limited Service, Free) Login

Search: The ACM Digital Library The Guide

Jcore <and> supertype <and> subtype <and> "interface"

THE GUIDE TO COMPUTING LITERATURE

Feedback Report a problem Satisfaction survey

Terms used **Jcore** and **supertype** and **subtype** and **interface** Found **582,505** of **827,115**

Sort results by: relevance

Display results: expanded form

Save results to a Binder

Search Tips

Open results in a new window

Try an Advanced Search

Try this search in The Digital Library

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next

Best 200 shown

Relevance scale

1 Languages: Support for subtyping and code re-use in Timor

J. Leslie Keedy, Gisela Menger, Christian Heinlein

February 2002 **Proceedings of the Fortieth International Confernece on Tools Pacific: Objects for internet, mobile and embedded applications - Volume 10**

Full text available: pdf(1.03 MB) Additional Information: full citation, abstract, references, index terms

Unlike most object oriented programming languages Timor, which has been designed to support component development, replaces the class construct with separate constructs for defining types and their implementations (which are not types). It also distinguishes between behaviourally conforming subtyping and the inclusion of behaviourally deviant interfaces in the definition of derived types. The separation of types and implementations simplifies a separation of subtyping and subclassing, facilitati ...

Keywords: base type, behavioural subtyping, code mapping, code re-use, derived type, polymorphism, software component, subclassing, subtyping, type

2 Polymorphism and subtyping in interface

Dinesh Katiyar, David Luckham, John Mitchell

August 1994 **ACM SIGPLAN Notices , Proceedings of the workshop on Interface definition languages**, Volume 29 Issue 8

Full text available: pdf(907.08 KB) Additional Information: full citation, abstract, citings, index terms

RAPIDE is a programming language framework designed for the development of large, concurrent, real-time systems by prototyping. The framework consists of a type language and default executable, specification and architecture languages, along with associated programming tools. *Interfaces* are the central construct of the type system. This document describes RAPIDE interfaces, their expressiveness and their use as a typing construct.

3 An extension of standard ML modules with subtyping and inheritance

John Mitchell, Sigurd Meldal, Neel Madhav

January 1991 **Proceedings of the 18th ACM SIGPLAN-SIGACT symposium on Principles of programming languages**

Full text available: pdf(1.10 MB) Additional Information: full citation, references, citings, index terms

4 Typechecking and modules for multimethods

Craig Chambers, Gary T. Leavens

November 1995 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 17 Issue 6

Full text available: pdf(2.90 MB) Additional Information: full citation, references, citings, index terms

Keywords: encapsulation, inheritance, multimethods, static typechecking, subtyping

5 Typechecking and modules for multi-methods

Craig Chambers, Gary T. Leavens

October 1994 **ACM SIGPLAN Notices , Proceedings of the ninth annual conference on Object-oriented programming systems, language, and applications**, Volume 29 Issue 10

Full text available: pdf(2.28 MB) Additional Information: full citation, abstract, references, citings, index terms

Two major obstacles hindering the wider acceptance of multi-methods are concerns over the lack of encapsulation and modularity and the absence of static typechecking in existing multi-method-based languages. This paper addresses both of these problems. We present a polynomial-time static typechecking algorithm that checks the conformance, completeness, and consistency of a group of method implementations with respect to declared message signatures. This algorithm improves on previous algori ...

6 On understanding types, data abstraction, and polymorphism

Luca Cardelli, Peter Wegner

December 1985 **ACM Computing Surveys (CSUR)**, Volume 17 Issue 4

Full text available: pdf(4.20 MB) Additional Information: full citation, abstract, references, citings, index terms, review

Our objective is to understand the notion of *type* in programming languages, present a model of typed, polymorphic programming languages that reflects recent research in type theory, and examine the relevance of recent research to the design of practical programming languages. Object-oriented languages provide both a framework and a motivation for exploring the interaction among the concepts of type, data abstraction, and polymorphism, since they extend the notion of type t ...

7 Behavioral contracts and behavioral subtyping

Robert Bruce Findler, Mario Latendresse, Matthias Felleisen

September 2001 **ACM SIGSOFT Software Engineering Notes , Proceedings of the 8th European software engineering conference held jointly with 9th ACM SIGSOFT international symposium on Foundations of software engineering**, Volume 26 Issue 5

Full text available: pdf(95.22 KB) Additional Information: full citation, abstract, references, citings, index terms, review

Component-based software manufacturing has the potential to bring division-of-labor benefits to the world of software engineering. In order to make a market of software components viable, however, producers and consumers must agree on enforceable software contracts. In this paper, we show how to enforce contracts if components are manufactured from class and interface hierarchies. In particular, we focus on one style of contract: pre- and post-conditions. Programmers annotate class and interface ...

8 Semantic database modeling: survey, applications, and research issues

Richard Hull, Roger King

September 1987 **ACM Computing Surveys (CSUR)**, Volume 19 Issue 3

Full text available: pdf(5.42 MB) Additional Information: full citation, abstract, references, citings, index terms, review

Most common database management systems represent information in a simple record-based format. Semantic modeling provides richer data structuring capabilities for database applications. In particular, research in this area has articulated a number of constructs that provide mechanisms for representing structurally complex interrelations among data typically arising in commercial applications. In general terms, semantic modeling complements work on knowledge representation (in artificial int ...

9 A language and environment for architecture-based software development and evolution

Nenad Medvidovic, David S. Rosenblum, Richard N. Taylor

May 1999 **Proceedings of the 21st international conference on Software engineering**

Full text available: pdf(1.56 MB) Additional Information: full citation, references, citings, index terms

Keywords: architecture description language, evolution, implementation mapping, software architecture, software environment

10 Specifications and their use in defining subtypes

Barbara Liskov, Jeannette M. Wing

October 1993 **ACM SIGPLAN Notices , Proceedings of the eighth annual conference on Object-oriented programming systems, languages, and applications**, Volume 28 Issue 10

Full text available: pdf(1.52 MB) Additional Information: full citation, references, citings, index terms

11 Type elaboration and subtype completion for Java bytecode

Todd B. Knoblock, Jakob Rehof

March 2001 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 23 Issue 2

Full text available: pdf(873.69 KB) Additional Information: full citation, abstract, references, citings, index terms

Java source code is strongly typed, but the translation from Java source to bytecode omits much of the type information originally contained within methods. Type elaboration is a technique for reconstructing strongly typed programs from incompletely typed bytecode by inferring types for local variables. There are situations where, technically, there are not enough types in the original type hierarchy to type a bytecode program. Subtype completion is a technique for adding necessary types to ...

Keywords: Java compiler, lattice completion, object-oriented type systems, type inference, type reconstruction, type-directed compilation, typed intermediate language

12 Correctly typing objects through subtyping and type inference

Xuequn Wu

March 1992 **Proceedings of the 1992 ACM/SIGAPP symposium on Applied computing: technological challenges of the 1990's**

Full text available: pdf(634.31 KB) Additional Information: full citation, references, citings, index terms

13 A type declaration and inference system for smalltalk

Alan H. Borning, Daniel H. H. Ingalls

January 1982 **Proceedings of the 9th ACM SIGPLAN-SIGACT symposium on Principles of programming languages**

Full text available: pdf(697.89 KB) Additional Information: full citation, abstract, references, citings

An experimental system for declaring and inferring type in Smalltalk is described. (In the current Smalltalk language, the programmer supplies no type declarations.) The system provides the benefits of type declaration in regard to compile-time checking and documentation, while still retaining Smalltalk's flexibility. A type hierarchy, which is integrated with the existing Smalltalk class hierarchy, allows one type to inherit the traits of another type. A type may also have parameters, which are ...

14 MultiJava: modular open classes and symmetric multiple dispatch for Java
Curtis Clifton, Gary T. Leavens, Craig Chambers, Todd Millstein
October 2000 **ACM SIGPLAN Notices , Proceedings of the 15th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications**, Volume 35 Issue 10

Full text available: pdf(196.92 KB) Additional Information: full citation, abstract, references, citings, index terms

We present MultiJava, a backward-compatible extension to Java supporting *open classes* and *symmetric multiple dispatch*. Open classes allow one to add to the set of methods that an existing class supports without creating distinct subclasses or editing existing code. Unlike the "Visitor" design pattern, open classes do not require advance planning, and open classes preserve the ability to add new subclasses modularly and safely. Multiple dispatch offers several well-known advantages ...

15 Type elaboration and subtype completion for Java bytecode
Todd B. Knoblock, Jakob Rehof
January 2000 **Proceedings of the 27th ACM SIGPLAN-SIGACT symposium on Principles of programming languages**

Full text available: pdf(1.48 MB) Additional Information: full citation, abstract, references, citings, index terms

Java source code is strongly typed, but the translation from Java source to bytecode omits much of the type information originally contained within methods. Type elaboration is a technique for reconstructing strongly typed programs from incompletely typed bytecode by inferring types for local variables. There are situations where, technically, there are not enough types in the original type hierarchy to type a bytecode program. Subtype completion is a technique for adding necessa ...

16 Keynote address - data abstraction and hierarchy
Barbara Liskov
January 1987 **ACM SIGPLAN Notices , Addendum to the proceedings on Object-oriented programming systems, languages and applications (Addendum)**, Volume 23 Issue 5

Full text available: pdf(1.38 MB) Additional Information: full citation, citings, index terms

17 Data abstraction and information hiding
K. Rustan M. Leino, Greg Nelson
September 2002 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 24 Issue 5

Full text available: pdf(469.27 KB) Additional Information: full citation, abstract, references, citings, index terms

This article describes an approach for verifying programs in the presence of data abstraction and information hiding, which are key features of modern programming languages with objects and modules. This article draws on our experience building and using an automatic program checker, and focuses on the property of *modular soundness*: that is, the property that the separate verifications of the individual modules of a program suffice to ensure the correctness of the composite program. We fo ...

Keywords: Abstract variables, abstraction dependencies, extended static checking, modifies clauses, modular verification, object-oriented programming, specifications

18 ADAM: a language-independent, object-oriented, design environment for modeling inheritance and relationship variants in Ada 95, C++, and Eiffel
D. Needham, S. Demurjian, K. El Guemhioui, T. Peters, P. Zamani, M. McMahon, H. Ellis
December 1996 **Proceedings of the conference on TRI-Ada '96: disciplined software development with Ada**

Full text available: pdf(1.45 MB) Additional Information: full citation, references, citings, index terms

19 A data modeling methodology for the design and implementation of information systems

Peter Lyngbaek, William Kent

September 1986 **Proceedings on the 1986 international workshop on Object-oriented database systems**

Full text available: pdf(981.90 KB) Additional Information: full citation, abstract, references, citings, index terms

Formal specifications that precisely and correctly define the semantics of software systems become increasingly important as the complexity of such systems increase. The emerging set of semantic data models which support both structural and operational abstractions are excellent tools for formal specifications. In this paper we introduce a methodology, based on an object-oriented data model, for the design and development of large software systems. The methodology is demonstrated by applyin ...

20 Changeable interfaces and promised messages for concurrent components

Franz Puntigam, Christof Peter

February 1999 **Proceedings of the 1999 ACM symposium on Applied computing**

Full text available: pdf(685.52 KB) Additional Information: full citation, references, citings, index terms

Keywords: interfaces, static type checking, subtyping

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next

The ACM Portal is published by the Association for Computing Machinery. Copyright © 2004 ACM, Inc.
Terms of Usage Privacy Policy Code of Ethics Contact Us

Useful downloads: Adobe Acrobat QuickTime Windows Media Player Real Player

Google

Web Images Groups News Froogle more »

D Language JCore Search Advanced Search Preferences

Web

Results 1 - 10 of about 166 for D Language JCore. (0.30 seconds)

[PDF] D: A Language Framework for Distributed Programming
File Format: PDF/Adobe Acrobat - View as HTML
... The component **language** in **D**, **Jcore**, is a subset of Java 1 . Additionally, **D** defines two small, specicalized, aspect **languages** that can be ignored for non ...
www2.parc.com/csl/groups/sda/ publications/papers/PARC-AOP-D97/ior-web.pdf - Similar pages

[PDF] D: A Language Framework for Distributed Programming
File Format: PDF/Adobe Acrobat - View as HTML
D: A **Language** Framework for Distributed Programming Cristina Videira Lopes ... independent **languages**: 1) one full object-oriented **language**, **JCore**, for programming ...
www2.parc.com/csl/groups/sda/ publications/papers/Lopes-Thesis/AppendixB.pdf - Similar pages
[More results from www2.parc.com]

D: A LANGUAGE FRAMEWORK FOR DISTRIBUTED PROGRAMMING
... The **D language** framework consists of: **Jcore**, an object-oriented **language** used to express the basic functionality and the activity of the system; ...
casl.csa.iisc.ernet.in/ProgrammingLanguages/D/tr-d.htm - 5k - Cached - Similar pages

gcc Information (c c++ compiler gcc gnu library linux software ...
... And our **Jcore** Library is tracking the evolving Java specification.As the choice ... codename "Bright" Project to create front end to gcc compiler for **D language**. ...
www.programming-x.com/programming/gcc.html - 24k - Cached - Similar pages

Separation of Concerns
... The **D language** framework consists of: (i) **Jcore**, an object-oriented **language** used to express the basic functionality and the activity of the system; (ii) Cool ...
www2.umassd.edu/SWSoC/papers.html - 73k - Cached - Similar pages

Alternative Java Implementations
... Dinkum® **Jcore** Library, Dinkumware – The Dinkum **Jcore** Library implements essential Java classes. ... You might also take a peek at the "**D**" **language**.
www.dwheeler.com/java-imp.html - 49k - Cached - Similar pages

Fate of Io: where's the code?
... **jcore** Active Days: 49. posted by **jcore** at Dec 4 ... extended as you are, and so I'**d** rather not ... in seeing (and possibly working on) the programming **language** that you ...
www.fateofio.org/v4/ v4.jsp?node=project&topic=1372&view - 12k - Cached - Similar pages

[PPT] CS696 Talk
File Format: Microsoft Powerpoint 97 - View as HTML
... 8. **D**. **Jcore** – component **language** (Java subset, removed synchronization). Cool – aspect **language** for expressing coordination of threads. ...
www.cs.virginia.edu/~evans/ cs655-S00/lectures/lecture21.ppt - Similar pages

[PPT] D A Framework for Distributed Programming
File Format: Microsoft Powerpoint 97 - View as HTML
... Aspectual Bloat. aspectual bloat =. LOC in Java - LOC in **JCore**. LOC in Cool+Ridl. ... **D**: Design. **D**: Implementation. Validation Results. ... 53. Directions in **Language** Design ...
www.ccs.neu.edu/research/demeter/ course/f99/lectures/lec6-RIDL.ppt - Similar pages

[PDF] April 4, 2003 1 JOHN E. CORE Office 2400 Steinberg-Dietrich Hall ...
File Format: PDF/Adobe Acrobat - View as HTML

Aspect-Oriented Logic Meta Programming (1998) (Make Corrections) (12 citations)
Kris De Volder
Lecture Notes in Computer Science

CiteSeer Home/Search Bookmark Context Related

View or download:
trese.cs.utwente.nl/aop...DeVolder.pdf
Cached: PS.gz PS PDF Image Update Help

From: emn.fr/sudholt/resea...references (more)
(Enter author homepages)

(Enter summary)

Rate this article: 1 2 3 4 5 (best)
Comment on this article

Abstract: It is our opinion that declaring aspects by means of a full-#edged logic language has a fundamental advantage over using a restricted special purpose aspect language. As an illustration we present a simpli#ed implementation of the Cool aspect weaver. Cool declarations are represented as logic facts in a Prolog like logic meta-language for Java. A fundamental advantage of this approach is that it enables aspect-orientedlogic meta programming. 1 Introduction We will brie#y introduce the... (Update)

Context of citations to this paper: More

...with the base language program using an aspect weaver. **In this area, the closest related work is the work by De Voider and D Hondt [15].** Their proposal utilizes a full fledged logic programming language as the aspect language. In order to specify concurrency issues in the...

...appropriate Java code automatically. **As it turned out, the approach was general enough to describe aspect oriented programming as well [DV99] Kim Mens uses the declarative meta programming approach in his dissertation [Men00] to verify the conformance of an...**

Cited by: More
Vrije Universiteit Brussel - Belgium - Faculty Of Sciences (Correct)
Concurrent Programming Made Easy - Ramirez, E.Santosa, H.C.Yap (2000) (Correct)
Using Inductive Logic Programming to Derive Software Views - Kellens (2003) (Correct)

Similar documents (at the sentence level):
22.2%: Type-Oriented Logic Meta Programming - De Volder (1998) (Correct)
6.6%: Aspect-Oriented Logic Meta Programming - De Volder, D'Hondt (1999) (Correct)

Similar documents based on text: More All
0.3: Type Oriented Logic Meta Programming for Java - De Volder (1998) (Correct)
0.2: Continuation-Passing-Style as an Intermediate Representation for .. - De Volder (1994) (Correct)
0.1: Aspect-Oriented Programming using a Logic Meta Programming.. - Gybels (2001) (Correct)

Related documents from co-citation: More All
5: Aspect-oriented programming - Kiczales, Lamping et al. - 1997
4: Inserting Ilities by Controlling Communications - Filman, Barrett et al.
4: A Logic Meta Programming Approach to Support the CoEvolution of Object-Oriented .. - Wuyts - 2001

BibTeX entry: (Update)

Kris De Volder. Aspect-Oriented Logic Meta Programming. Position paper at the ECOOP '98 workshop on Aspect-Oriented Programming. http://citeseer.ist.psu.edu/devolder98aspectoriented.html More

```
@article{ devolder99aspectoriented,
    author = "Kris De Volder and Theo D'Hondt",
    title = "Aspect-Oriented Logic Meta Programming",
    journal = "Lecture Notes in Computer Science",
    volume = "1616",
    pages = "250--??",
    year = "1999",
    url = "citeseer.ist.psu.edu/devolder98aspectoriented.html" }
```

Citations (may not include all citations):

CiteSeer Find: JCore programming language Documents Citations

Searching for PHRASE **jcore programming language**.
Restrict to: Header Title Order by: Expected citations Hubs Usage Date Try: Google (CiteSeer) Google (Web) CSB DBLP
No documents match Boolean query. Trying non-Boolean relevance query.
500 documents found. **Only retrieving 250 documents (System busy - maximum reduced). Order: relevance to query.**

Nets, Logic and Concurrent Object-Oriented Programming - Rafael Ramirez (Correct)
Nets, logic and concurrent object-oriented **programming** Rafael Ramirez Dept. of Computer Science
Tempo, a declarative concurrent **programming language**, is based on such logic and has a procedural
and so on. Several concurrent logic **programming languages** have been proposed (e.g. Parlog [Gregory 87]
www.cs.bris.ac.uk/~ramirez/Petri.ps

Encapsulated Search and Constraint Programming in Oz - Christian Schulte, Gert.. (1994) (Correct) (21 citations)
Encapsulated Search and Constraint **Programming** in Oz Christian Schulte, Gert Smolka, and
to create a high-level concurrent **programming language** providing the problem solving capabilities of
It exploits the fact that Oz is a higher-order **language**. The search combinator is given an expression E
ftp.cs.washington.edu/pub/constraints/ppcp94/schulte.ps.Z

Flexible Exception Handling in the OPERA Process Support System - Hagen, Alonso (1997) (Correct) (12 citations)
is bound to be a normal occurrence. Any **programming** tool intended for large, complex applications
support system using a combination of **programming language** concepts and transaction processing techniques.
its main ideas from a combination of **programming language** concepts and transaction processing techniques,
www.inf.ethz.ch/personal/alonso/PAPERS/ah98-icdcs98.ps.Z

A New Programming Paradigm for Engineering Design Software - Salustri, Venter (1994) (Correct)
A New **Programming** Paradigm for Engineering Design Software F.
a prototype-based object-oriented **programming language** supporting a signature-based canonical message
is implemented using the Scheme **programming language**. Because Designer satisfies a formal conceptual
salustri.esxf.uwindsor.ca/~fil/Papers/designer-ewc94.ps

Object Oriented Parallel Programming Experiments and Results - Lee, Gannon (1991) (Correct) (30 citations)
Object Oriented Parallel **Programming** Experiments and Results Jenq Kuen Lee Dennis
distributed collection model and an experimental **language** PCbased on this model. In the distributed
of **programming** and an experimental **programming language** and a compiler project that we hope can be used
faicon.cs.nthu.edu.tw/~jklee/papers/long_supercomp.ps.Z

Generating Efficient Code for Lazy Functional Languages - Smetsers, Nöcker, van.. (1991) (Correct) (20 citations)
Clean, an experimental lazy functional **programming language**, code is generated for an
Generating Efficient Code for Lazy Functional **Languages** Sjaak Smetsers, Eric Nocker, John van
code generator can be built for (lazy) functional **languages**. Starting from Concurrent Clean, an
ftp.cs.kun.nl/pub/CSI/SoftwEng.FunctLang/papers/smes91-codegeneration.ps.gz

Multi-Language Programming Environments for High Performance Java.. - Getov (1999) (Correct) (2 citations)
Multi-**Language Programming** Environments for High Performance Java
Multi-**Language Programming** Environments for High Performance
capabilities, software tools, **programming languages** and **programming** paradigms have brought about
www.cs.cf.ac.uk/hpjworkshop/vladimir.ps

Drafting ER and OO Schemas in Prototyping Environments - Meyer, Westerman, Gogolla (1996) (Correct)
Model Object-Oriented Modelling Logic **Programming** Prototyping. 1 Introduction Conceptual
semantically well-founded query and manipulation **language** based on an EntityRelationship calculus. The
data model, and in its support of an expressive **language** for queries and constraints. All features
www.db.informatik.uni-bremen.de/publications/Meyer_1996_DKE.ps.gz

Induction, Logic, and Natural Language Processing - Luc Dehaspe (Correct)
knowledge, the cross-fertilization between logic **programming** and machine learning has given rise to a new
Induction, Logic, and Natural **Language** Processing Luc Dehaspe, Hendrik Blockeel, and
of logic **programming** within both natural **language** research and machine learning, we point out
www.cs.kuleuven.ac.be/~ldh/PAPERS/mlnlp96.ps.gz

Collection-Oriented Languages - Sipelstein (1991) (Correct) (50 citations)
or the U.S. government. Abstract Several **programming languages** arising from widely diverse
Collection-Oriented **Languages** Jay M. Sipelstein Guy E. Blelloch March 18,
U.S. government. Abstract Several **programming languages** arising from widely diverse practical and
www.cs.cmu.edu/~scandal/papers/CMU-CS-90-127.ps.gz

Multi-Paradigm Programming through Graph Rewriting: Case for.. - Glauert (Correct)
Multi-Paradigm **Programming** through Graph Rewriting: Case for Support Dr.
for practical implementations of multi-paradigm **languages**, based on largely declarative principles. The
based on largely declarative principles. The **languages** chosen will combine process and functional
www.sys.uea.ac.uk/~jrwg/MultiPar/MultiParCase.ps.gz

On Implementations and Semantics of a Concurrent Programming.. - Sewell (1997) (Correct) (4 citations)
On Implementations and Semantics of a Concurrent **Programming Language** Peter Sewell 1 Abstract The
and Semantics of a Concurrent **Programming Language** Peter Sewell 1 Abstract The concurrency theory
application of the calculus, the **programming language** Pict of Pierce and Turner, primarily in order to
www.cl.cam.ac.uk/users/pes20/pict9-crc11pt.ps.gz

A Classical Linear lamda-calculus - Bierman (1997) (Correct)
level, certain control operators for functional **programming languages** [23]To some extent Girard's linear
the paradigm of control operators for functional **languages** which gives a new computational interpretation
control operators for functional **programming languages** [23]To some extent Girard's linear logic [21]
sunsite.doc.ic.ac.uk/ic.doc/theory/guests/BiermanGM/cll.ps.gz

TAM Technical Notes 1 and 2 - David Scholefield (Correct)
which is closer to that of a typical imperative **programming language** than the original, rather terse,
for TAM A new syntax for the formal real-time **language** TAM is presented which is closer to that of a
to that of a typical imperative **programming language** than the original, rather terse, notation. In
www.cs.york.ac.uk/ftpdir/reports/YCS-93-213.ps.Z

Beauty and the Beast or A Formal Semantic Description .. - Baumann, Fässler.. (1993) (Correct) (1 citation)
developed has been implemented in the functional **programming language** Standard ML, thereby resulting in an
paper we provide the denotational semantics of a **language** we call MiCo 1 containing the essential
Cobol. MiCo can therefore be regarded as a core **language** of Cobol. The semantics we have developed has
ftp.ifi.unizh.ch/pub/institute/baumann/bab.ps.Z

Navigating the MeldC - The Meldc (1991) (Correct)
c fl1992 Howard Gershen, Erik Hilsdale The **Programming** Systems Laboratory is supported by National
Hseush (Doctoral Candidate) Principal designer of **language** kernel Co-designer of compiler James Lee
Winfrey 8 CONTENTS Introduction The MeldC **language** was developed at Columbia University to bring
www.cs.columbia.edu/~library/TR-repository/reports/reports-1991/cucs-031-91.ps.gz

A Visual Approach To Distributed Programming - Ng, Kramer, Magee, Dulay (1996) (Correct) (1 citation)
1 A Visual Approach To Distributed **Programming** Keng Ng, Jeff Kramer, Jeff Magee And Naranker
component types described in a **programming language**. From these, the executing distributed system
for defining the semantics of our configuration **language**. More recently, we have been working on the use
dse.doc.ic.ac.uk/dse-papers/design/kluwer.ps.gz

Code Composition as an Implementation Language for Compilers - Stichnoth, Gross (1997) (Correct) (7 citations)
system. 1 Introduction Application-specific **programming languages** capture properties of a particular
Code Composition as an Implementation **Language** for Compilers James M. Stichnoth and Thomas
high-level operations. The developer (i.e.the **language** designer or compiler writer) provides building
pecan.srv.cs.cmu.edu/afs/cs.cmu.edu/user/stichnot/public/www/dsl97.ps

Extending A General-Purpose Algebraic Modeling Language To.. - Robert Fourer (1998) (Correct) (4 citations)
Language To Combinatorial Optimization: A Logic **Programming** Approach Robert Fourer Department Of
Extending A General-Purpose Algebraic Modeling **Language** To Combinatorial Optimization: A Logic
General-purpose algebraic modeling **languages** are a central feature of popular computer
iems.nwu.edu/~4er/WRITINGS/loglang.ps.gz

A Formal Framework for Modules With State - Ancona, Zucca (1996) (Correct) (1 citation)
within an imperative context. A module in a **programming language** is a part of a program viewed as a
an imperative context. A module in a **programming language** is a part of a program viewed as a whole (often
hidden to users (encapsulation)A modular **language** supports a variety of ways for composing
ftp.disi.unige.it/pub/person/AnconaD/AMAST96.ps.gz

CiteSeer Find: supertype and subtype and UML Documents Citations

Searching for **supertype and subtype and uml**.
Restrict to: Header Title Order by: Expected citations Hubs Usage Date Try: Google (CiteSeer) Google (Web) CSB DBLP
20 documents found. **Order: number of citations.**

An Integrated Metamodel for OCL Types - Baar, Hähnle (2000) (Correct) (5 citations)
1.3 [4]according to [5]the type OclAny is the **supertype** of all types while [4, p. 7-29] states that
\t]he prede ned OCL Collection types are not **subtypes** of OclAny"the type metamodel in [5]
Abstract The main objectives of OCL are to restrict **UML** models by additional constraints and to clarify
i12www.ira.uka.de/~baar/publications/2000/baarhaehnle00.ps.gz

A Comparison of Role Mechanisms in Object-Oriented.. - Kappel, Retschitzegger.. (1998) (Correct) (4 citations)
operators for viewing an object in its roles as a **supertype** and a **subtype**, respectively, are called coerce-
may be extended to an instance of one or several **subtypes** of this type. An object having several such
modeling methods, such as OMT [Rumb91] and **UML** [Fowl97]do not require that an object is
ftp.ifs.uni-linz.ac.at/pub/publications/1997/1397.ps.gz

Subtyping Patterns for Active Objects - Heike Wehrheim Universitat (2000) (Correct) (3 citations)
a **subtype** object should be usable wherever a **supertype** object is expected. Even stronger, we also want
requirement imposed on types and their **subtypes**. For active objects with an associated behaviour
formal methods. One witness to this is certainly the **UML** combining traditional class diagrams with a number
wwwmath.uni-muenster.de/cs/u/versys/workshops/VVVNS2000/papers/Wehrheim.pdf

Best of Both Worlds - A Mapping from EXPRESS-G to UML - Arnold, Podehl (1998) (Correct) (2 citations)
through an open circle. The EXPRESS-G **supertype-subtype** relation symbol is a thick line with a
through an open circle. The EXPRESS-G **supertype-subtype** relation symbol is a thick line with a circle
Best of Both Worlds -A Mapping from EXPRESS-G to **UML** Florian Arnold, Gerd Podehl Research Group for
sumatra.my.uni-kl.de/publikationen/Best_of_Both_Worlds_LNCS.pdf

Improving the UML Metamodel to Rigorously Specify Aggregation and .. - Barbier (2001) (Correct) (1 citation)
of the more general concept of WPR. The WPR **supertype** is defined by clearly enunciated
Aggregation and Composition are treated as **subtypes** of the more general concept of WPR. The WPR
Improving the **UML** Metamodel to Rigorously Specify Aggregation and
www.univ-pau.fr/~messine/R2I/R2I01-02.ps

Merging Collection Data Structures in a Content Management System - Wienberg (2003) (Correct)
boolean 1 declares target **subtype** 0.1 **supertype** *MediaProperty StringProperty
maxCard: int duplicates: boolean 1 declares target **subtype** 0.1 **supertype** *MediaProperty StringProperty
the CRP is object-oriented, implementing a subset of **UML**. Objects are the semantic unit for reference,
www.sts.tu-harburg.de/papers/2003/Wien03.pdf

Checking Behavioural Subtypes - Via Refinement Heike (Correct)
we expect the **subtype** to be substitutable for the **supertype** while at the same moment allowing extension of
Checking Behavioural **Subtypes** Via Refinement Heike Wehrheim Universit-At
increasing importance with the spreading use of **UML** in system design. In **UML**, a behaviour description
csd.informatik.uni-oldenburg.de/~wehrheim/fmoods02.ps

Evolving Partitions in Conceptual Schemas in the UML - Gómez, Olivé (2002) (Correct)
between entity types E' the **subtype**) and E (the **supertype**) means that if an entity e is instance of E'
Man and Woman states that Man and Woman are **subtypes** of Person, that Man and Woman are disjoint, and
Evolving Partitions in Conceptual Schemas in the **UML** Cristina Gmez and Antoni Oliv Departament de
www.mm.di.uoa.gr/~rouvas/ssi/caise2002/23480467.pdf

The Rdf Distillery - Gibson (Correct)
.19 2.14 The **subtype/supertype** relationship definition in config.xml.
. 19 2.14 The **subtype/supertype** relationship definition in config.xml.
homepage. 29 3.4 **UML** class diagram of ValidateRDF.
www.kvocentral.com/reports/diongibsonthesis.pdf

Incremental Specialization vs. Overriding Specialization.. - Pons, Perez, Giandini (2002) (Correct)
A **subtype** is completely compatible with its **supertype**, that means the **subtype** has the same argument

were expected. When subclasses do not imply **subtypes** this kind of subsumption may lead to wrong
At the present the Unified Modeling Language (**UML**) is considered the standard modeling language for
www.matcom.uh.cu/eventos/ideas2002/articulos/completos/IncrementalSpecializacion/IncrementalSpecialization.PDF

Life-Cycle Inheritance in Concurrent Object-Oriented Systems - Philippi (Correct)
which has no relation to the dynamic model of its **supertype**. This leads to the situation that instances of
Consequently, a dynamic model may be assigned to a **subtype** which has no relation to the dynamic model of
due to the advent of notational frameworks like the **UML** (OMG 2001) and others widely accepted today. In
www.jrpit.flinders.edu.au/confpapers/CRPITV12Philippi.ps

Types for Multi-Paradigm Modelling - de Lara, Vangheluwe (Correct)
of the defined type, and determining **subtype/supertype** relationships. Keywords: Modelling &
variables of the defined type, and determining **subtype/supertype** relationships. Keywords: Modelling &
modelling formalisms Entity-Relationship Diagrams, **UML** class Diagrams Meta-Model Model used to specify
www.CS.McGill.CA/~hv/publications/02.IFIP.article.pdf

Ontological Engineering for Conceptual Modeling - Raban, Garner (Correct)
P, it can be said that a property P is a (proper) **supertype** of a property Q. There could be two different
in time t. 2.1 Subtyping If a property Q is a **subtype** of a property, denoted by Q P, then t (Q
and integration very difficult. XML [2]XMI [3] or **UML** [4]5]6]to mention just some of the
sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-48/Raban.pdf

Comprehensive Specification of Distributed Systems Using - Bastarrica, Demurjian.. (2000) (Correct)
subtype has the same interfaces and calls as its **supertypes** and it extends them. In Figure 2, we show the
inheritance is also specified at this level: a **subtype** has the same interfaces and calls as its
The graphical notation is based on customized **UML** implementation diagrams [5]and the textual
www.dcc.uchile.cl/~cecilia/./papers/scco2000.ps

View Integration of Behavior in Object-oriented Databases - Preuner, Conrad, Schrefl (Correct)
both object types during integration, a common **supertype** will be formed that contains the common parts
contains the common parts of the behaviors of its **subtypes**, i.e.a type hierarchy is defined in the
Some objectoriented models (e.g.OMT [18] and **UML** [3]use state charts [9] for modeling behavior
www.dke.uni-linz.ac.at/research/publications/TR0001.ps.gz

UML data models from an ORM perspective: Part 8 - Halpin (1999) (Correct)
the constrained **subtypes** collectively exhaust the **supertype**, but this latter notion is far more useful in
that the modeler intended to add no more **subtypes**. This weak notion of completeness does not
UML data models from an ORM perspective (part 8) 1
www.orm.net/pdf/orm-jcm8.pdf

UML data models from an ORM perspective: Part 6 - By Dr Terry (Correct)
instance of a **subtype** is also an instance of its **supertype**(s)For example, declaring Woman to be a
(is-a"semantics, where each instance of a **subtype** is also an instance of its **supertype**(s)For
UML data models from an ORM perspective (part 6) 1
www.orm.net/pdf/orm-jcm6.pdf

Ad/98-07-09 - July Omg Stream-Based (Correct)
0 AUmlGeneralizationSupertype 24 Generalization **supertype** Generalizable Element 0 CUmlInterface 24
24 AUmlGeneralizationSubtype 24 Generalization **subtype** Generalizable Element 0
Model Interchange Format Page 31 of 42 Appendix A: **UML** Core Package Mapping to SMIF Architecture Subject
ftp.omg.org/pub/docs/ad/98-07-09.pdf

Try your query at: Google (CiteSeer) Google (Web) CSB DBLP

CiteSeer - Copyright NEC and IST